A Ranking Model for Software Requirements Prioritization during Requirements Engineering: a case study

Ishaya P. Gambo
Department of Computer Science and Engineering, Faculty of Technology
Obafemi Awolowo University
Ile-Ife, Nigeria
ipgamb@gmail.com

Rhoda N. Ikono
Department of Computer Science and Engineering, Faculty of Technology
Obafemi Awolowo University
Ile-Ife, Nigeria
rhoda_u@yahoo.com

Philip O. Achimugu
Department of Computer Science, Lead City University, Ibadan, Nigeria
check4philo@gmail.com

Olaronke G. Iroju
Department of Computer Science, Adeyemi College of Education, Ondo,Nigeria
rojuolaronke@gmail.com

***Abstract*- Software requirements prioritization is a recognized practice in requirements engineering (RE) that facilitates the management of stakeholders' subjective views as specified in their requirements listing. Since RE process is naturally collaborative in nature, the intensiveness from both knowledge and human perspectives opens up the problem of decision making on requirements, which can be facilitated by requirements prioritization. However, due to the large volume of requirements elicited when considering an ultra-large-scale system, existing prioritization techniques proposed so far suffer some setbacks in terms of efficiency, effectiveness and scalability. This paper employed the use of a more efficient ranking algorithm for requirements prioritization based on the limitations of existing techniques. The major objective is to provide a well-defined ranking procedure through analysis, suitable for prioritizing software requirements. An empirical evaluation of the proposed technique was made using a typical scenario of the Pharmacy Information System at the Obafemi Awolowo University Teaching Hospital Complex (OAUTHC) as a case study. The results showed the computation of the positive ideal solution (PIS) and negative ideal solution (NIS), as well as the closeness coefficient (CC) for 4 requirements across 3 stakeholders. The CC showed the final ranks of requirements, where R4 with 2.09 point is the most valued requirements, while R1 and R2 with CC of 1.37 and 1.05 were next in the order of priority respectively. The CC provides the medium through which problems of multiple criteria decision making can be handled, so as to determine the order of priority of the available alternatives. The paper conveyed encouraging evidence for the software engineering community that is capable of resolving redundant specified requirements, thereby providing the potential that will facilitate effective and efficient decision making in handling the differences amongst requirements that have been prioritized. Thus, prioritizing software requirements with the recommended ranking procedure during software development is crucial and vital in order to reduce development cost.**

Keywords: *Software systems, requirements engineering, requirements prioritization, ranking algorithm.*

I. INTRODUCTION

In engineering software systems during requirements engineering process, requirements prioritization is essential for the purpose of implementing an agreeably ultra-large scale system. For instance, in developing critical systems with large number of stakeholders' requirements, prioritization can help in facilitating the choice of the final requirements listing as specified by the stakeholders [1]. Thus, prioritization of elicited requirements is essential in the development of software products that will meet the desired goals of stakeholders. The requirements in this context include useful information that will satisfy the need of the users or project stakeholders [2], and prioritizing them will help to prevent breaches in contracts such as budget over-shoot, exceeding delivery time and missing out important requirements during implementation [3]. We therefore see requirements prioritization as a process of managing the subjective views of stakeholders as specified in their requirements listing. This is with the aim of handling and negotiating the contradictory and conflicting expectations from each stakeholder among other reasons specified in Liaskos et al [4].

However, the selection and prioritization of requirements for the purpose of engendering a system of high quality is seen as a major challenge in software development [5]. It is obvious in literature that prioritization supports the recognition of all the foremost requirements as perceived by relevant stakeholders [6], and it is the activity required for the selection of appropriate requirements [7]. This is with a view of implementing the core sets of requirements with respect to cost, quality, available resources and delivery time [8, 9]. Hence, this paper further buttressed the importance of requirements prioritization and suggests that unmistakable ranking of requirements is an essential success factor for ensuring efficient requirements engineering process. In this case, the importance of every requirement should be based on each stakeholder's subjective view, which makes it multidimensional since it is dependent on the stakeholders' perspective [10]. The primary objective of ranking stakeholders' requirements in a software development process is to aid the analysis of the to-be system by providing the order of their implementation plan amidst available alternatives [11, 12, 13, 14]. However, due to the large volume of data (referred to as stakeholders' requirements), a number of prioritization techniques proposed so far suffer some setbacks. The resultant effect of such setbacks makes it impossible for software developers to unveil interesting and actionable information about the expected requirements for the software project and product.

In this paper, we propose a well suited ranking procedure given the basic flow of processes expected in the prioritization of software requirements, and the corresponding algorithm. The remaining part of this article has been structured as follows: The second section reveals the related works, the third section deals with the proposed method indicating the various tasks of the ranking procedures and emphasized on its suitability and relevance for software prioritization. The fourth section presents the experimental set-up also covering the empirical evaluation of proposed method on a case study; the fifth section presents and discussess the results; while the sixth section concludes the research and identify suggested future work.

II. Related works

In the software engineering literatures, several authors support the need for prioritizing software requirements in making the right choice from the different viewpoint aspects of stakeholders. For instance, the authors in [15, 16, 17, 18, 19] considered how requirements can be prioritized in a multi-team agile context, considering the change-driven nature of the agile methodology. Even in the goal oriented requirements engineering (GORE) methodology, prioritization is considered most important for the purpose of picking out the goals with respect to domain specific needs [20, 21, 22, 23]. However, providing a well-defined ranking procedure suitable for prioritizing software requirements across the various aspects and methodology is essential for implementation plan amidst available alternatives.

Consequently, the advantages of prioritizing software requirements is beneficial to software development project and practises as established for decades in the literature. For example, Pitangueira *et al.* [24] conducted a systematic review to investigate and analyze the various approaches proposed in literature to address software prioritization and selection problems. Their emphasis was based on Search-Based Software Engineering (SBSE), and their findings indicated the aspects addressed by most researchers, and the most prominent techniques used based on the defined problem. Additionally, Gambo et al. [1] considered the possibility of integrating Fuzzy Multi Criteria Decision Making (FMCDM) alongside similarity measures and target-based approaches to requirements prioritization using linguistic values of triangular fuzzy numbers. The emphasis in [1] was on how to avert subsequent system failure by making precise and accurate decision in developing large scale software systems. However, an algorithm that will rank and/order these sets of requirements is essential to the enhancement of the proposed techniques in [1].

Other examples of related works from literature include: (1) the work of Babar *et al.* [5] on the analysis of various issues associated with existing prioritization techniques. These authors observed that existing prioritization techniques suffer a lot of setback because they can only handle software projects with very few requirements specifications. Consequently, this has rendered current techniques unsuitable for the prioritization of large scale sets of requirements during software development. (2) the work of Achimugu *et al.* [25] provided another in-depth review that was based on the classification of existing prioritization techniques, and focused on their various setbacks and processes. The authors made some pertinent discoveries and recommendations on the grey areas for enhancement. (3) the work of Pergher and Rossi [26] provided the justification of evaluating prioritization tools by conducting a methodological mapping study. This was further supported by Dabbagh *et al.* [27] with focus on executing two consecutive controlled experiments that aimed at evaluating current prioritization techniques. (4) the work of Riņķevičs and Torkar [28] that focused on the empirical analysis of commutative voting and the corresponding outcome. The authors proposed the ECV methods for the analysis of results from the CV technique. Again, this technique suffers some setback in terms of analyzing, negotiating and prioritizing large number of requirements during development.

In most cases, the common result with prioritizing software requirement is an ordering of prioritized lists of requirements that needs to be considered first during the software development process [29]. The justification for the acceptance of software systems by its stakeholders have been based on how well the requirements are captured, analyzed and prioritized [30, 31, 32]. The literature contained a lot of contribution that have been made in requirements prioritization research [33, 34]. This is evident in the renowned number of techniques that have been proposed and implemented at different times, and on different development projects. Common to these techniques is their ability to define requirements with greater value to business successes. Racheva *et al* [35] and Berander *et al.* [36] provided different categorization of prioritization techniques.

The first categorization by Racheva *et al.* [35] includes two main classes: (1) techniques applicable to small-scale requirements, for example, round-the-group prioritization, multi-voting system, pair-wise analysis, weighted criteria analysis, and the quality function deployment techniques; and (2) techniques useful for large-scale requirements, for example, MoSCoW, binary priority list, planning game, case based rank and the Wiegers's matrix techniques.

The second categorization by Berander *et al.* [36] also provided two main classifications. The first classification was based on techniques that support the assignment of values and/or weights by project stakeholders on each requirement. The essence of this classification is to describe the importance of each requirement comparatively. An example of this includes the analytical process (AHP), planning game, cumulative voting, numerical assignment, and Wieger's method. The second classification suggests the approaches supporting negotiation. In this case, the priorities of requirements are ascertained from the agreement established among the subjective evaluation given by each stakeholder. An example of this classification is the Win-Win model and multi criteria preference analysis requirement negotiation (MPARN) technique.

However, all the techniques from each categorization given in [35] and [36] have their limitations. For example, with the Analytical Hierarchy Process (AHP) as a technique, an $n \times (n-1)/2$ comparison was proposed at each hierarchy level given n requirements. This has been evidently seen as a shortcoming of the AHP. This is because when the number of requirements increases, the number of comparisons will also increase with a magnitude of $O(n^2)$ [37, 38]. In addition, the AHP and CBRanks techniques have demonstrated high capabilities as reported in [5]. Both techniques are easy to use, and their accuracy level is very high. Still, their limitations are based on lack of support for scalability [39, 40, 41], and the inability to support rank reversals. This is obvious when considering a larger number of requirements for developing large scale system like the hospital systems. Detailed work that provided more insight on existing requirements prioritization and their corresponding shortcomings have been reported in [25] and other related works. Therefore, we proposed a more efficient ranking algorithm for requirements prioritization based on the limitations of existing techniques.

III. PROPOSED METHOD

Fig. 1 described the procedures and corresponding processes adapted from Perini et al. [42] for the purpose of prioritizing software requirements. It consists of two phases which involve the manual and automated phases. The manual phases have to do with the specification and weighting of requirements while the automated process has to do with ranking of the weighted requirements and the display of the final ranked requirement. As shown in fig. 1, the ranking procedure indicating the basic flow of processes involved in prioritizing software requirements consists of some important tasks which must be observed in order to achieve reliable prioritization results. These tasks are enumerated below:

1) Requirement Elicitation: This is the number one task in the process of requirements prioritization. It involves the elicitation, gathering and/or extraction of requirements. To elicit requirements, the elicitor, developer or engineer must articulate the description of problem source and how the proposed software is meant to solve the problem identified to the co-opted stakeholders. These stakeholders must acquire adequate understanding of the problem domain and proposed solution before elicitation commences. In cases where stakeholders come with their own requirements, adequate care must also be taken to analyze each requirement with respect to feasibility of implementation, relevant hardware support availability and compatibility, availability of skilled programmers, budget constraints, delivery time and the overall value of the proposed software to the organization or end-users. To initiate the elicitations process, the stakeholders are led to express their views in short sentences where each stakeholder quietly document requirements. Thereafter, stakeholders engage in a round-robin feedback to collate all the requirements so that a discussion section could be organized to arrive at consensus requirements and resolve ambiguities. The requirement elicitation process is given by Equation 1 below:

$$R_{(s,d)} = \min \sum_{i=1}^{d} max \sum_{j=1}^{d} \alpha \mathrm{R}_{(s_i,d_j)} \begin{cases} \alpha = 1\,; \\ \alpha = \mathrm{n}\,; \end{cases} \qquad (1)$$

Where:

- i and j are the specified requirements from the first to the last stakeholder, respectively.
- $\mathrm{R}_{(s,d)}$ is the total number of elicited requirements from an origin node, *"s"* to a last node, *"d"*.
- α is an integer which is non-negative
- $\mathrm{R}_{(s_i,d_j)}$ is the total number of stakeholders and their respective specified requirements.

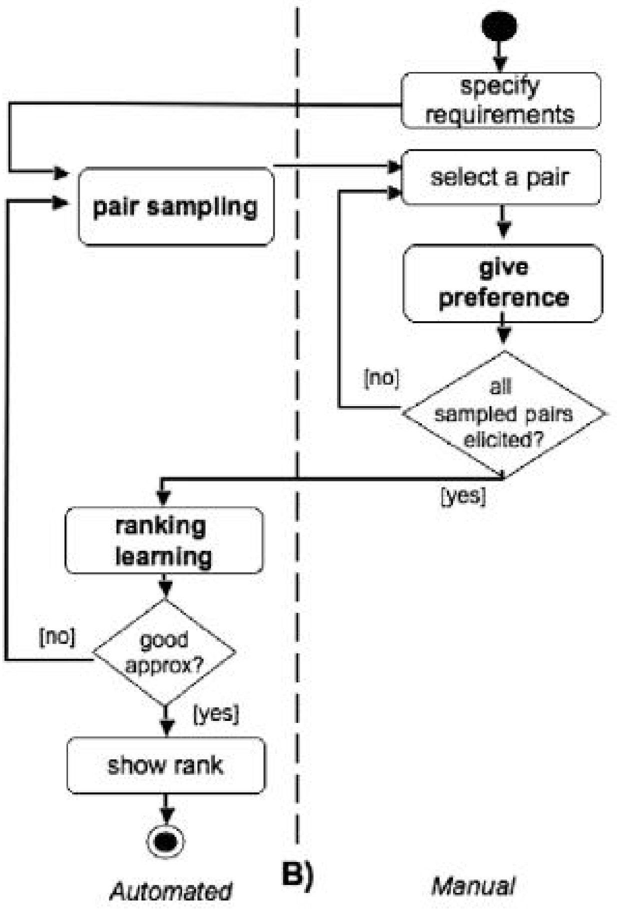

Fig. 1. Model's Information Flow (Adapted from Perini *et al.* [42])

2) Pair Sampling: This task is required in determining the relative importance of a requirement against the other. The relative importance is determined based on pre-defined criteria. Usually, the criteria are set based on the targeted output of the proposed system. Given some sets of requirements (*ri, rj*∈ R), the process of pair sampling can be represented with the sets of equations enumerated below:

$$r_{(j-1,j)} \otimes r_{j-1}, r_{(i,j)} = W_j \text{ ; s.t. } i,j \in R \quad (2)$$

$$r_{(j-2,j-1)} \otimes r_{j-2}, r_{(i-1,j-1)} = W_{j-1} \text{ ; s.t. } i,j \in R \quad (3)$$

$$r_{(j-3,j-2)} \otimes r_{j-3}, r_{(i-2,j-2)} = W_{j-2} \text{ ; s.t. } i,j \in R \quad (4)$$

$$\ldots \otimes \ldots, \ldots = \ldots \text{ ; s.t. } i,j \in R$$

$$\ldots \otimes \ldots, \ldots = \ldots \text{ ; s.t. } i,j \in R$$

$$r_{(1,2)} \otimes W_1, r_{(2,3)} = W_2 \text{ ; s.t. } i,j \in R \quad (5)$$

Where:

- *i* and *j* are the consensus requirements from the first to the last
- $\otimes$ is the transitive operator that aid the comparison between one requirement and the other
- r are given sets of requirements
- R is the pool of consensus requirements

3) Preference Elicitation: This is the act of obtaining results of the pair sampling processes from the stakeholders. More precisely, this task is concerned with the collection of all the weighted requirements based on set criteria (*c*). These weights are eventually used to compute of the final rank of the entire requirements. The preference elicitation process is executed with the following expressions:

$\otimes : R \times R \rightarrow \{-1, 0, 1\}$ where $\otimes$ $(r_i, r_j)\ c = 1$ means that r_j has been ranked above r_i,

$\otimes$ $(r_i, r_j)c = -1$ means that r_i has been ranked above r_j, and

$\otimes$ $(r_i, r_j)c = 0$ indicates that no preference has been given between r_i and r_j

(We assume $\otimes$ $(r_i, r_j)\ c = 0$ and $\otimes$ $(r_i, r_j)c = -\otimes$ $(r_j, r_i)\ c$ for all $r_i, r_j \in R$).

4) Ranking Learning: The weights of requirements obtained from (iii) serves as input during ranking learning. The idea here is to compute all the specific ranks of each requirement across all the stakeholders. The learning process relies on the prescribed weights

and ranking criteria to process ranking results. Given some sets of weights; W_i, W_j, on requirements (r_i, $r_j \in$ R); ranking R^* can be executed using the equation 6:

$$R^*_{(s,d)} = \sum_{i=1}^{d} W_i\, ri, W_j rj \sum_{j=1}^{d} W_n\, rn, W_m rm R^*_{(s_i,d_j)}$$

s.t. i & j; n &m ≥ 1; (6)

Where:

- R^* stands for ranked requirements
- *s* and *d* are denoted for the start and end of the consensus requirements
- *W* stands for the weights allotted to each requirement
- r are given sets of requirements
- *i* and *j* represents the relative requirements
- *n* and *m* represents the end of relative requirements

5) The Final Rank: This is the output of the entire process which reflects the value of each requirement as perceived by the stakeholders. Software requirement specifications deal with the representations of structural components and the relationships that tie them together. In order to avoid vague specification of requirements, we have suggested the application of composition operator ($\otimes$) that will support pair wise comparison which has the capacity of pre-processing requirements. The synchronization of these representations can be easily achieved if all the components of the proposed software are well identified and consistently articulated. The preceding description of the proposed requirement prioritization technique forms part of the research contributions and will lead to unambiguous requirements specification.

The inputs to the model consist of the following:

(a) A set of finite requirements $R = \{r_1, \ldots, r_n\}$.
(b) The ranking criteria $C = (c_1, \ldots, c_m)$.
(c) The stakeholders' preferences represented by the function $\otimes$ $(r_i, r_j)\ c$

During rank computations, the essence is to generate a priority list of all the requirements contained in *R*. This priority is represented by the function P*:* $R \rightarrow$ R. The function *P* stands for the hierarchy of *R* determined by the preference weights allotted to the various requirements. The aim of any prioritization exercise is to reduce the level of discrepancies or disagreements between prioritized requirements in order to curb ranking error as well as ensure scalable prioritization process. Therefore, if r_i, r_j, are given sets of requirements, it is important to ensure the transitive closure $\otimes$ $(r_i, r_j) > 0$. This will ensure non-zero weights on any requirement. The proposed algorithm has the capacity of ordering requirements based on weighted scores.

In fig. 2, we demonstrated that the actual input to the ranking process is a finite set of elicited requirements that are about to be ranked while the ultimate output of the entire process consists of sets of displayed ranked requirements based on pre-defined criteria. The criteria in this context outlined the steps considered suitable in the ranking process as defined in the ranking algorithm.

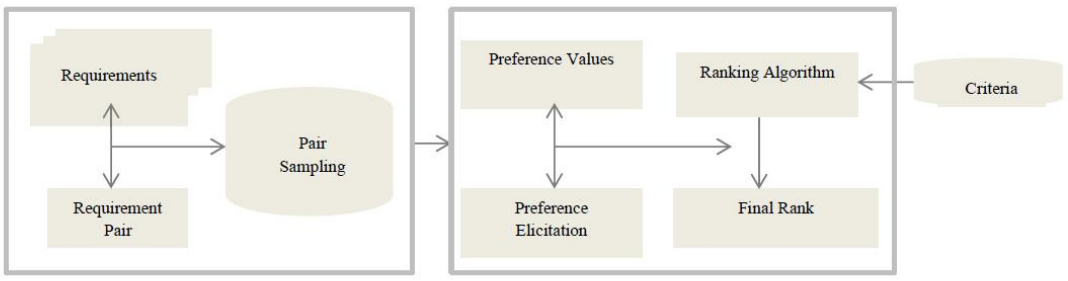

Fig. 2. Ranking-based flow of processes in prioritizing software requirements

In the algorithm, the focus here is to assign membership functions against each requirement based on the allotted weights in order to construct a decision matrix. For instance, assuming we have sets of requirements *R* elicited from a particular software development project, $R_i(i = 1,2,\ldots,m)$ which is been evaluated with respect to the number *n* of selected criteria $C_j(j = 1,2,\ldots,n)$ that constitute each of those requirements; then, weights can be determined by relevant stakeholders to track the following: (a) the criteria that are to be utilized in ranking the requirements and (b) construction of the decision matrix using the linguistic variables expressed in Table 1.

TABLE I: Linguistic variables

S/N	Variables	Meaning
1	VL	Very Low
2	L	Low
3	M	Medium
4	H	High
5	VH	Very High

The weighted criteria W stand for the comparative rank of the chosen criteria; while, decision matrix represents the overall ranking of each requirement R_i in line with the prescribed criteria C_j. After obtaining the weighted requirements W; the proposed algorithm works as follows:

Step 1: Construction of the decision matrix X: The weights are used to construct decision matrix of the form shown below:

$$D = \begin{array}{c} \\ R_1 \\ R_2 \\ \vdots \\ R_m \end{array} \begin{array}{c} \begin{array}{cccc} c_1 & c_2 & \cdots & c_n \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \end{array} \qquad (7)$$

$$W = [\, w_1 \quad w_2 \quad \cdots \quad w_n \,]$$

After constructing decision matrix, there is need to normalize the matrix by using Equation 8.

$$R_{ij} = \sum_{j=1}^{i} x_{ij} i = 1,...n; j = 1,..m \qquad (8)$$

Step 2: Aggregation of normalized decision weights: The aggregated weights for the normalized entries are computed by obtaining the square root of each of the normalized weight using Equation 9.

$$W_j = \sqrt[m]{w_{1,j} ... w_{m,j}}, \qquad j = 1, \ldots, n \qquad (9)$$

Step 3: Computation of the aggregated fuzzy decision matrix in its linguistic form using Equation 10.

$$R_{ij} = \sum_{j=1}^{i} x^2{}_{ij} i = 1,...n; j = 1,..m \qquad (10)$$

Step 4: Computation of the global weights: The *ith* values in Step 3 are multiplied by each values of the aggregated decision matrix in Step 2 using Equation 11 to acquire the global weights of each criterion.

$$W_j = \sqrt[m]{w_{1,j} \times w_{2,j} \times w_{3,j} ... w_{m,j}}, \times \sum_{j=1}^{i} x^2 ij, i = 1,..n; j = 1,...m \qquad (11)$$

Step 5: Computation of the Positive Ideal Solution (PIS) and Negative Ideal Solution (NIS): The solutions from the global decision matrix values in Step 4 are used to compute the PIS and NIS. To compute the PIS, Equation 12 is utilized.

$$A^* = \left(v_1{}^*, v_2{}^*, \cdots, v_n{}^*\right) \qquad (12)$$

Where, v_j^* = the maximum values of the requirement entries in the aggregated decision matrix. NIS is computed with Equation 13.

$$A' = \left(v_1', v_2', \cdots, v_n' \right) \qquad (13)$$

Where, v' = the minimum values of the requirement entries in the aggregated decision matrix.

Step 6: Computation of the separating distances: This is achieved by finding the variance amidst the highest and lowest values contained in the aggregated decision matrix as depicted in Equation 14.

$$S_i = \left(\sum (A^*_{ij} - A^-_{ij}) \right) \qquad (14)$$

Where $i = 1,2,\cdots,m$

Step 7: The results in step 6 represent the confidence rating of each requirement with respect to the relevant stakeholders. Therefore, the requirement with the highest confidence rating (CR) value is considered as the prime requirement.

In dealing with decision making challenges during software elicitation process, it is necessary to consider the number of requirements specified by stakeholders (or the number of criteria that makes up each requirement) in order to find the Euclidean distances amongst positively ideal requirements. Assuming a particular software engineer have acquired two major requirements for a certain software development project, say *(R_1, R_2)*; then it becomes lighter to determine the PIS criteria which consist of all highly ranked criteria that constitute a requirement. Conversely, the NIS which consists of criteria that are least ranked is also detected. However, if two requirements R_1 and R_2 possess a shorter distance to both PIS and NIS; it becomes quite imperative to clearly define the rationale for choosing one over the other or choosing both as the case may be. This algorithm deals with the concept of considering alternatives, known as compromise solution, that possesses the nearest distance to the PIS and the farthest distance from the NIS.

IV. Experimental set-up

To illustrate the concept of these techniques, the Pharmacy Information System at the Obafemi Awolowo University Teaching Hospital Complex (OAUTHC) was considered in this case. Consider the following scenario: “The Pharmacy Department at OAUTHC would like to develop new software to replace the existing one so that a better solution for the pharmacy operation is achieved, and each pharmacy sub-unit’s functions and activities are captured. The new system should facilitate the administration of both outpatient and inpatient medication supplies to the wards, and also provides inventory support to manage stock movement from any given location across the three tiers of healthcare delivery system in Nigeria”.

The system must be flexible enough to enable Pharmacists and other experts gain access into the system and administer the appropriate healthcare services. Bearing in mind that, OAUTHC and other hospitals deal with complex and large amount of data; there would be a need to build a scalable system that will be dynamic and interoperable. Thus, the need to provide an appropriate measure of ranking requirements in a manner to avoid delay in implementation, reduce cost of development, and ensure quality assurance and control. The system should also be able to assist in patient care by the monitoring of drug interactions, drug allergies and other possible medication-related complications, in addition to capturing relevant medical information. The essence of capturing this medical information of patients could be aimed at indexing into a well-structured database that is void of redundancy or replications. Basically, the software system can be developed to enable Pharmacist to manage medication orders, ensure that preparations, dispensing, and verification are all easily completed and monitored.

From the above scenario, the software engineers and other relevant stakeholders are trying to gather information in order to ensure that, the proposed system works as expected. Nine stakeholders are involved in this case. Based on the elicited information, the architectural design decisions are made. Due to several reasons, architectural decision-making is a difficult task because: (i) requirements are usually captured with a lot of ambiguous insinuations (ii) functional and non-functional requirements are difficult to explicitly specify (iii) concrete architectural decisions have to be made based on vague requirements in some cases (iv) stakeholders involved have different perspectives, views or concerns.

Possessing a mastery of sound elicitation technique and being able to clearly describe problems leads to concise specification of requirements. This is the key for developing a formidable architecture for software systems. To get started, software engineers and stakeholders must be able to choose views, sort them, prioritize and document them which translate to the following sequence of activities:

a. The utilization of an elicitation technique must lead to unambiguous requirements specification.
b. The architectural design must conform to the elicited requirements.
c. The quality attributes of the proposed system are selected from the non-functional requirements class.
d. The mapping between architectural design decisions and requirement specifications must be consistent.

During or after requirements specification, it is important to collapse them into categories as described in the hierarchical representation of the specified requirements shown in fig. 3. This will enhance clarity in the evaluation process; since these requirements are about to be prioritized. When prioritization is to commence, requirements are compared pair-wisely where relevant stakeholders are required to determine the relative importance of each requirement based on the comparison scale. The requirements undergoing pair wise comparison are performance, reusability, flexibility and maintainability; denoted as R_1, R_2, R_3 and R_4 respectively.

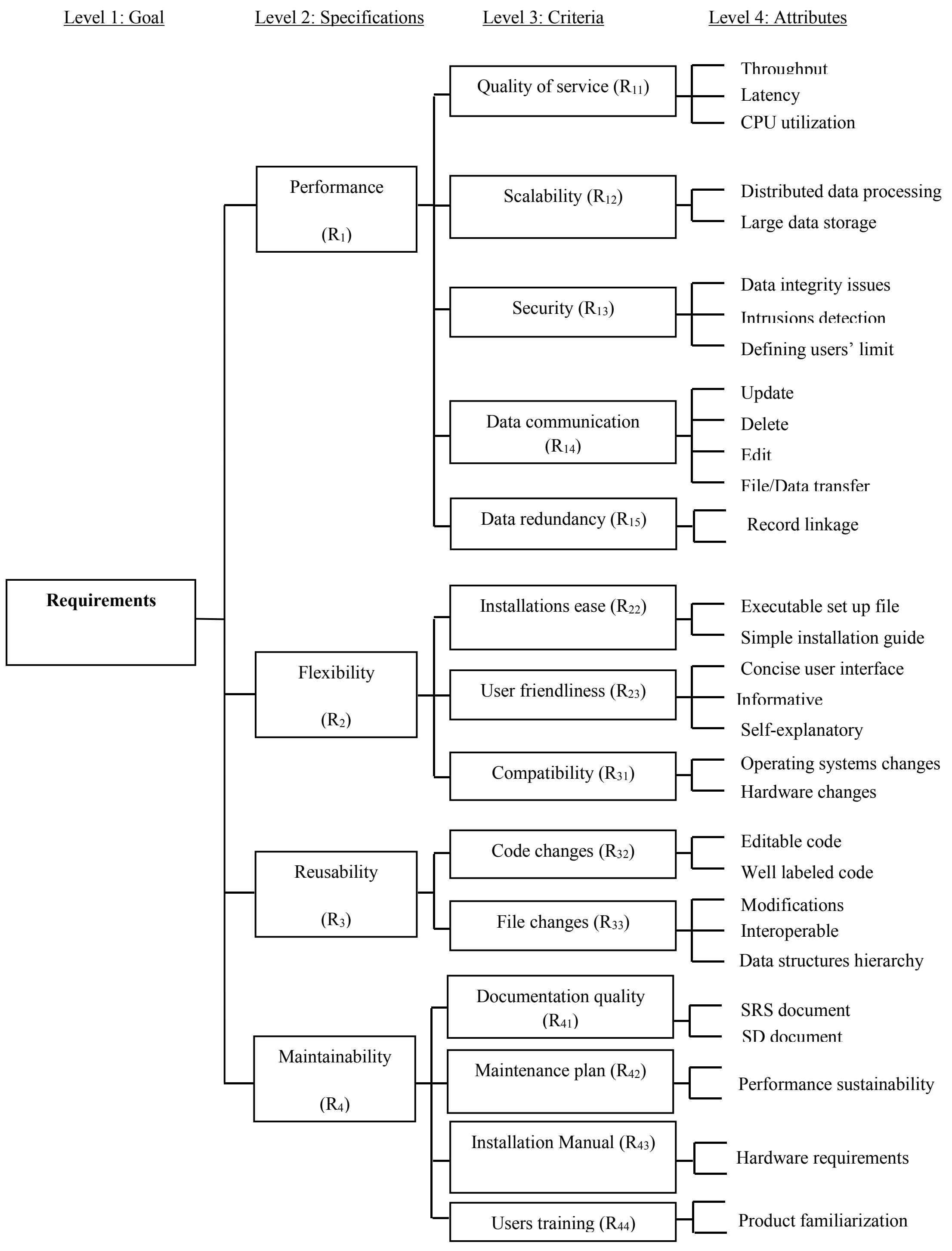

Fig. 3. Categorization of Stakeholder's requirements

In the context of the Pharmacy information system from literature, performance has to do with the overall deliveries supported by a system. However, the attributes that constitute this requirement include quality of service (system availability); scalability (large amount of data processing at runtime) and Security (protection of patient data against malicious or unauthorized users), data communication (basic database updates across distributed database applications) and data redundancy (ability of a distributed database system to detect or avoid data replications by linking records to appropriate entities).

The requirement tagged flexibility on the other hand has to do with the level of portability of the system. That is, the ability of the system to be installed and used across any operating system platform. Precisely, the attributes under this category are installation ease (ability for the system to be easily installed on independent platforms); user friendliness (a system with concise and unambiguous features or functions). Interfaces are expected to be designed with clear links or Webpages and compatibility; that is, the ability of the system to adapt to sudden changes in technological or infrastructural advancement.

Reusability measures the extent at which a system can be long lived; that is, the degree at which a system can evolve or adapt to operating system changes or modifications, as well as data, algorithm and file changes. This is crucial because, a system that is not reusable will not have any future value. If a system ends up not having any future value, it creates an impression that money has been wasted in the project. Moreover, organisations in recent times are beginning to appreciate systems that possess the ability to adapt to future changes with little or no administrative interventions.

Finally, the maintenance requirement is responsible for prolonging or sustaining the life span of the developed software systems. This involves the provision of the software requirement and design documents as well as the application codes, a periodic maintenance plan and a user training session to enable users get themselves acquainted with the software system and installation manual which will help guide users to a successful installation of the software. This could also help them perform simple maintenance exercises.

V. RESULTS AND DISCUSSION

Table 2 indicates how the subjective priorities given to each requirement by the stakeholders were captured. To achieve that, the linguistic valuations were introduced. In this case, Ranking Scales (RS) were used as variables. The RS are assertions that use expressions in English language to represent values that stands for the degree of acceptability of a particular variable or parameter. One advantage of the linguistic variables is the ability to determine the level of acceptability amongst specified requirements by each stakeholder. Therefore, using the RS, stakeholders were able to provide preference weights for the elicited requirements based on their perceived importance.

Next to the priority list indicated by the RS is Tables 3 and 4 showing the ranking with normalised and aggregate weights derived from equation 8 and 9 respectively. These normalised decision weights determine the performance of ranking algorithm. Tables 5 and 6 showed the Fuzzy and Global decision matrices derived from equation 10 and 11 respectively. The fuzzy logic concept was employed in [1] as a way of overcoming complexities in decision making. Consequently, equation 12 was used to compute the positive ideal solution (PIS) and negative ideal solution (NIS), as well as the closeness coefficient (CC).

The two ideal solutions (PIS and NIS) were computed for 4 requirements across 3 stakeholders as shown in Tables 7 and 8 respectively. PIS and NIS was introduced to provide intuitive and computationally feasible approach to identify the CC. Hence, the CC showing the final ranks of requirements are depicted in Table 9. The CC provides the medium through which problems of multiple criteria decision making can be handled, so as to determine the order of priority of available alternatives. Experimentally, the CC showed that R4 with 2.09 point is the most valued requirements. Next to R4 in the order of priority is R1 and R2 respectively with CC of 1.37 and 1.05 respectively. Thus, we are optimistic that this ranking algorithm can cater for large dataset (i.e. large number of requirements). As such, prioritizing software requirements with the suggested ranking algorithm during software development can reduce development cost, and aid smooth release management of software products timely.

Therefore, the results in Tables 2 to 9 represent the performance of the proposed technique. These results emphasized more on scalability and ranking evolving requirements. The advantages of the ranking algorithm are as follows:

(a) Scalability: The algorithm can be applied to requirements of ultra-large scale software development projects since the weights of requirements can be computed across all their respective criteria at runtime.
(b) Evolvability (Rank reversal): The algorithm also has the capacity of accurately calculating weights across even or odd numbers of criteria that constitute each requirement. The functions for computing PIS and NIS have the capacity of computing relative weights between odd or even requirements.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a more efficient ranking algorithm that is capable of resolving redundant specified requirements. The ranking algorithm proposed, demonstrated the potential of catering for large number of requirements as datasets. Secondly, it has the potential of facilitating effective and efficient decision making that can handle the differences amongst requirements that have been prioritized. Therefore, it is believed that this approach can help software engineers to prioritize requirements capable of forecasting the expected behaviour of software under development. The implication of the final ranked requirements would be that those that emerged top in the ranked list will be implemented first while others can be implemented subsequently in the software release planning process.

The next phase of this research can be tailored towards the minimization of discrepancies between ranked requirements. Secondly, the automation of the computations proposed in this paper and the display of ranked results will be required. Finally, it will be so interesting to validate this proposed technique with a real life software development project and applicable datasets as well as implementation of the prototype.

TABLE II: Linguistic variables

R_1	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
S_1	VH	VH	VH	VH	VH
S_2	VH	VH	VH	VH	VH
S_3	H	H	H	H	H
S_4	H	H	H	H	H
S_5	VH	VH	VH	VH	VH
S_6	H	H	H	H	H
S_7	VH	VH	VH	VH	VH
S_8	H	H	H	H	H
S_9	VH	VH	VH	VH	VH

R_2	R_{21}	R_{22}	R_{23}
S_1	H	H	H
S_2	VH	VH	VH
S_3	VH	VH	VH
S_4	M	M	M
S_5	H	H	H
S_6	VH	VH	VH
S_7	VH	VH	VH
S_8	M	M	M
S_9	H	H	H

R_3	R_{31}	R_{32}
S_1	H	H
S_2	VH	VH
S_3	VH	VH
S_4	VH	VH
S_5	M	M
S_6	M	M
S_7	M	M
S_8	H	H
S_9	M	M

R_4	R_{41}	R_{42}	R_{43}	R_{44}
S_1	VH	VH	VH	VH
S_2	VH	VH	VH	VH
S_3	M	M	M	M
S_4	VH	VH	VH	VH
S_5	H	H	H	H
S_6	H	H	H	H
S_7	H	H	H	H
S_8	H	H	H	H
S_9	VH	VH	VH	VH

TABLE III: Normalized weights

R_1	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
S_1	0.10	0.10	0.10	0.10	0.10
S_2	0.10	0.10	0.10	0.10	0.10
S_3	0.06	0.06	0.06	0.06	0.06
S_4	0.06	0.06	0.06	0.06	0.06
S_5	0.10	0.10	0.10	0.10	0.10
S_6	0.06	0.06	0.06	0.06	0.06
S_7	0.10	0.10	0.10	0.10	0.10
S_8	0.06	0.06	0.06	0.06	0.06
S_9	0.10	0.10	0.10	0.10	0.10

R_2	R_{21}	R_{22}	R_{23}
S_1	0.06	0.06	0.06
S_2	0.10	0.10	0.10
S_3	0.10	0.10	0.10
S_4	0.02	0.02	0.02
S_5	0.06	0.06	0.06
S_6	0.10	0.10	0.10
S_7	0.10	0.10	0.10
S_8	0.02	0.02	0.02
S_9	0.06	0.06	0.06

R_3	R_{31}	R_{32}
S_1	0.06	0.06
S_2	0.10	0.10
S_3	0.10	0.10
S_4	0.10	0.10
S_5	0.02	0.02
S_6	0.02	0.02
S_7	0.02	0.02
S_8	0.06	0.06
S_9	0.02	0.02

R_4	R_{41}	R_{42}	R_{43}	R_{44}
S_1	0.10	0.10	0.10	0.10
S_2	0.10	0.10	0.10	0.10
S_3	0.02	0.02	0.02	0.02
S_4	0.10	0.10	0.10	0.10
S_5	0.06	0.06	0.06	0.06
S_6	0.06	0.06	0.06	0.06
S_7	0.06	0.06	0.06	0.06
S_8	0.06	0.06	0.06	0.06
S_9	0.10	0.10	0.10	0.10

TABLE IV: Aggregate weights

R_1	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}
S_1	0.316	0.316	0.316	0.316	0.316
S_2	0.316	0.316	0.316	0.316	0.316
S_3	0.245	0.245	0.245	0.245	0.245
S_4	0.245	0.245	0.245	0.245	0.245
S_5	0.316	0.316	0.316	0.316	0.316
S_6	0.245	0.245	0.245	0.245	0.245
S_7	0.316	0.316	0.316	0.316	0.316
S_8	0.245	0.245	0.245	0.245	0.245
S_9	0.316	0.316	0.316	0.316	0.316

R_2	R_{21}	R_{22}	R_{23}
S_1	0.245	0.245	0.245
S_2	0.316	0.316	0.316
S_3	0.316	0.316	0.316
S_4	0.141	0.141	0.141
S_5	0.245	0.245	0.245
S_6	0.316	0.316	0.316
S_7	0.316	0.316	0.316
S_8	0.141	0.141	0.141
S_9	0.245	0.245	0.245

R_3	R_{31}	R_{32}
S_1	0.245	0.245
S_2	0.316	0.316
S_3	0.316	0.316
S_4	0.316	0.316
S_5	0.141	0.141
S_6	0.141	0.141
S_7	0.141	0.141
S_8	0.245	0.245
S_9	0.141	0.141

R_4	R_{41}	R_{42}	R_{43}	R_{44}
S_1	0.316	0.316	0.316	0.316
S_2	0.316	0.316	0.316	0.316
S_3	0.141	0.141	0.141	0.141
S_4	0.316	0.316	0.316	0.316
S_5	0.245	0.245	0.245	0.245
S_6	0.245	0.245	0.245	0.245
S_7	0.245	0.245	0.245	0.245
S_8	0.245	0.245	0.245	0.245
S_9	0.316	0.316	0.316	0.316

TABLE V: Fuzzy decision matrix

R_1				R_2				R_3				R_4			
S_1	0.80	1.25	1.25	S_1	0.27	0.48	0.75	S_1	0.18	0.32	0.50	S_1	0.64	1.00	1.00
S_2	0.80	1.25	1.25	S_2	0.48	0.75	0.75	S_2	0.32	0.50	0.50	S_2	0.64	1.00	1.00
S_3	0.45	0.80	1.25	S_3	0.48	0.75	0.75	S_3	0.32	0.50	0.50	S_3	0.16	0.36	0.64
S_4	0.45	0.80	1.25	S_4	0.12	0.27	0.48	S_4	0.32	0.50	0.50	S_4	0.64	1.00	1.00
S_5	0.80	1.25	1.25	S_5	0.27	0.48	0.75	S_5	0.08	0.18	0.32	S_5	0.36	0.64	1.00
S_6	0.45	0.80	1.25	S_6	0.48	0.75	0.75	S_6	0.08	0.18	0.32	S_6	0.36	0.64	1.00
S_7	0.80	1.25	1.25	S_7	0.48	0.75	0.75	S_7	0.08	0.18	0.32	S_7	0.36	0.64	1.00
S_8	0.45	0.80	1.25	S_8	0.12	0.27	0.48	S_8	0.18	0.32	0.50	S_8	0.36	0.64	1.00
S_9	0.80	1.25	1.25	S_9	0.27	0.48	0.75	S_9	0.08	0.18	0.32	S_9	0.64	1.00	1.00

TABLE VI: Global fuzzy decision matrix

R_1				R_2				R_3				R_4			
S_1	1.00	1.57	1.57	S_1	0.26	0.47	0.73	S_1	0.13	0.22	0.35	S_1	0.72	1.12	1.12
S_2	1.00	1.57	1.57	S_2	0.47	0.73	0.73	S_2	0.25	0.40	0.40	S_2	0.72	1.12	1.12
S_3	0.50	0.89	1.38	S_3	0.47	0.73	0.73	S_3	0.25	0.40	0.40	S_3	0.12	0.27	0.48
S_4	0.50	0.89	1.38	S_4	0.15	0.26	0.47	S_4	0.25	0.40	0.40	S_4	0.72	1.12	1.12
S_5	1.00	1.57	1.57	S_5	0.26	0.47	0.73	S_5	0.04	0.10	0.17	S_5	0.36	0.63	0.99
S_6	0.50	0.89	1.38	S_6	0.47	0.73	0.73	S_6	0.04	0.10	0.17	S_6	0.36	0.63	0.99
S_7	1.00	1.57	1.57	S_7	0.47	0.73	0.73	S_7	0.04	0.10	0.17	S_7	0.36	0.63	0.99
S_8	0.50	0.89	1.38	S_8	0.15	0.26	0.47	S_8	0.13	0.22	0.35	S_8	0.36	0.63	0.99
S_9	1.00	1.57	1.57	S_9	0.26	0.47	0.73	S_9	0.04	0.10	0.17	S_9	0.72	1.12	1.12

TABLE VII: Positive ideal solution

				d^+
R_1	1.00	1.57	1.57	4.14
R_2	0.47	0.73	0.73	1.93
R_3	0.25	0.40	0.40	1.05
R_4	0.72	1.12	1.12	2.96

TABLE VIII: Negative ideal solution

				d^-
R_1	0.50	0.89	1.38	2.77
R_2	0.15	0.26	0.47	0.88
R_3	0.04	0.10	0.17	0.31
R_4	0.12	0.27	0.48	0.87

TABLE IX: Closeness coefficient

	d^+	d^-	CC	*Ranking*
R_1	4.14	2.77	1.37	2
R_2	1.93	0.88	1.05	3
R_3	1.05	0.31	0.74	4
R_4	2.96	0.87	2.09	1

Acknowledgements

The Health Informatics Research Group (HIRG) of Computer Science and Engineering, Faculty of Technology, Obafemi Awolowo University, Ile-Ife, Nigeria is hereby acknowledged for some of the facilities utilized during the course of this research.

Reference

[1] I. Gambo, R. Ikono, P. Achimugu, and A. Soriyan, “An Integrated Framework for Prioritizing Software Specifications in Requirements Engineering,” *International Journal of Software Engineering and its Applications (IJSEIA)*, Vol. 12, No. 1, 2018, pp. 33-46. ISSN 1738-9984.

[2] P. Achimugu, and A. Selamat, “A Hybridized Approach for Prioritizing Software Requirements Based on K-Means and Evolutionary Algorithms,” *Computational Intelligence Applications in Modelling and Control,* 2015, pp. 73-93. Springer International Publishing.

[3] P. Achimugu, A. Selamat, and R. Ibrahim, “ReproTizer: A Fully Implemented Software Requirements Prioritization Tool,” *Transactions on Computational Collective Intelligence*, XXII, 2016, pp. 80-105. Springer Berlin Heidelberg.

[4] S. Liaskos, S. A. McIlraith, S. Sohrabi, and J. Mylopoulos, “Representing and reasoning about preferences in requirements engineering,” *Requirements Engineering*, Vol. 16, No. 3, 2011, pp. 227-249.

[5] M. I. Babar, M. Ghazali, D. N. Jawawi, S. M. Shamsuddin, and N. Ibrahim, “PHandler: An expert system for a scalable software requirements prioritization process,” *Knowledge-Based Systems*, Vol. 84, 2015, pp. 179-202.

[6] G. Ruhe, A. Eberlein, and D. Pfahl, “Trade-off Analysis for Requirements Selection,” *International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, No. 4, 2003, pp. 345-366.

[7] R. K. Chopra, V. Gupta, and D. S. Chauhan, “Experimentation on Accuracy of Non Functional Requirement Prioritization Approaches for different complexity Projects,” *Perspectives in Science,* 2016, http://dx.doi.org/10.1016/j.pisc.2016.04.001.

[8] A. Finkelstein, M. Harman, S. Mansouri, J. Ren, and Y. Zhang, “A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making,” *Requirements Engineering*, Vol. 14, 2009, pp. 231–245.

[9] S. Barney, A. Aurum, and C. Wohlin, “A product management challenge: creating software product value through requirements selection,” *Journal of System Architecture*, Vol. 54, 2008, pp. 576–593.

[10] P. Belsis, A. Koutoumanos, and C. Sgouropoulou, “PBURC: a patterns-based, unsupervised requirements clustering framework for distributed agile software development,” *Requirements Engineering*, Vol. 19, No. 2, 2014, pp. 213-225.

[11] A. S. Jadhav, and R. M. Sonar, “Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach,” *Journal of Systems and Software,* Vol. 84, No. 8, 2011, pp. 1394-1407.

[12] L. Pareto, A. B. Sandberg, P. Eriksson, and S. Ehnebom, "Collaborative prioritization of architectural concerns," *Journal of Systems and Software,* Vol. 85, No. 9, 2012, pp. 1971-1994.

[13] P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," *Information and software technology*, Vol. 55, No. 1, 2013, pp. 173-187.

[14] J. Karlsson, S. Olsson, and K. Ryan, "Improved practical support for large-scale requirements prioritizing," *Requirements Engineering*, Vol. 2, No. 1, 1997, pp. 51-60.

[15] A. De Lucia, and A. Qusef, "Requirements engineering in agile software development," *Journal of Emerging Technologies in Web Intelligence*, Vol. 2, No. 3, 2010, pp. 212-220.

[16] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal*, Vol. 20, No. 5, 2010, pp. 449-480.

[17] M. Daneva, E. Van Der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, and R. Wieringa, "Agile requirements prioritization in large-scale outsourced system projects: An empirical study," *Journal of systems and software*, Vol. 86, No. 5, 2013, pp. 1333-1353.

[18] Z. Bakalova, M. Daneva, A. Herrmann, and R. Wieringa, "Agile requirements prioritization: What happens in practice and what is described in literature," In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality* (REFSQ 2011), 28th - 30th March, Essen, Germany, 2011, pp. 181-195. Springer, Berlin, Heidelberg.

[19] I. Inayat, S.S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in human behavior*, Vol. 51, 2015, pp. 915-929.

[20] S. Vinay, S. Aithal, and G. Sudhakara, "A quantitative approach using goal-oriented requirements engineering methodology and analytic hierarchy process in selecting the best alternative," In *Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI), 22 Aug - 25 Aug 2013, Mysore, India,* 2013, pp. 441-454.

[21] M. Sadiq, and S. K. Jain, "Stakeholder identification method in goal oriented requirements elicitation process," In *proceedings of the IEEE 5th International Workshop on Requirements Prioritization and Communication (RePriCo, 2014), 26-26 August, Karlskrona, Sweden,* 2014, pp. 25-33.

[22] M. Sadiq, and S. K. Jain, "A fuzzy based approach for the selection of goals in goal oriented requirements elicitation process," *International Journal of System Assurance Engineering and Management*, Vol. 6, No. 2, 2015, pp. 157-164.

[23] M, Sadiq, T. Hassan, and S. Nazneen, AHP_GORE_PSR: applying analytic hierarchy process in goal oriented requirements elicitation method for the prioritization of software requirements. In *the Proceedings of the IEEE 3rd International conference on Computational Intelligence & Communication Technology (CICT, 2017), 9th - 10th February, Ghaziabad, India,* 2017, pp. 1-5.

[24] A. M. Pitangueira, R. S. P. Maciel, and M. Barros, "Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature," *Journal of Systems and Software*, Vol. 103, 2015, pp. 267-280.

[25] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. R. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, Vol. 56, 6, 2014, pp. 568-585.

[26] M. Pergher, and B. Rossi, "Requirements prioritization in software engineering: a systematic mapping study," In: *Paper presented at the 2013 IEEE Third International Workshop on Empirical Requirements Engineering (EmpiRE),* 2013.

[27] M. Dabbagh, S. P. Lee, and R. M. Parizi, "Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches," *Soft Computing,* 2015, pp. 1-24.

[28] K. Riņķevičs, and R. Torkar, "Equality in cumulative voting: A systematic review with an improvement proposal," *Information and Software Technology,* Vol. 55, No. 2, 2013, pp. 267-287.

[29] M. Dabbagh, and S. P. Lee, "An approach for prioritizing NFRs according to their relationship with FRs," *Lecture Notes on Software Engineering,* Vol. 3, No. 1, 2015, pp. 1-5.

[30] A. M. Davis, "*The art of requirements triage*", IEEE Computer, Vol. 36, No. 3, 2003, pp. 42-49.

[31] N. R. Mead, "Requirements Prioritization Introduction," *Software Engineering Institute Web Publication, Carnegie Mellon University, Pittsburgh, USA,* 2006.

[32] A. Perini, F. Ricca, A. Susi, and C. Bazzanella, "An empirical study to compare the accuracy of AHP and CB Ranking techniques for requirements prioritization," in: *Proceedings of the Fifth International Workshop on Comparative Evaluation in Requirements Engineering, IEEE, 2007*, pp. 23–35.

[33] T. L. Saaty, The analytic hierarchy process, McGraw-Hill, New York, 1980.

[34] A. Herrmann, and M. Daneva, "Requirements prioritization based on benefit and cost prediction: an agenda for future research," In: *RE, IEEE Computer Society*, 2008, pp. 125–134.

[35] Z. Racheva, M. Daneva, A. Herrmann, and R. J. Wieringa, “A conceptual model and process for client-driven agile requirements prioritization,” In: *IEEE (2010) Fourth International Conference on Research Challenges in Information Science (RCIS),* 2010, pp. 287–298.

[36] P. Berander, K. A. Khan, and L. Lehtola, “Towards a research framework on requirements prioritization,” *SERPS 6*, 2006, pp. 18–19.

[37] J. R. Hubbard, “*Theory and Problems of Data Structures with C++*”. McGraw-Hill, NY, USA, 2000.

[38] N. W. Kassel, and B. A. Malloy, “An approach to automate requirements elicitation and specification,” In *the Proceedings of 7th International Conference on Software Engineering and Applications,* 2003, pp. 3-5.

[39] J. Karlsson, and K. Ryan, “*Cost-value approach for prioritizing requirements*” IEEE Software, Vol. 14, No. 5, 1997, pp. 67-74.

[40] J. Karlsson, C. Wohlin, and B. Regnell, “An evaluation of methods for prioritizing software requirements,” *Information and Software Technology*, Vol. 39, No. 14-15, 1998, pp. 939-947.

[41] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, “Towards automated requirements prioritization and triage, Requirements Engineering,” Vol. 14, No. 2, 2009, pp. 73–89.

[42] A. Perini, F. Ricca, and A. Susi, “Tool-supported requirements prioritization: Comparing the AHP and CBRank methods,” *Information and Software Technology*, Vol. 51, No. 6, 2009, pp. 1021-1032.